

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-168145

(43) Date of publication of application : 14.06.1994

(51)Int.Cl.

G06F 9/46

G06F 12/00

(21) Application number : 03-352297

(71)Applicant : HITACHI LTD

(22) Date of filing : 13.12.1991

(72)Inventor : SHIMIZU TAKESHI
TAKEYAMA HIROSHI
KOBAYAKAWA MANABU

(30) Priority

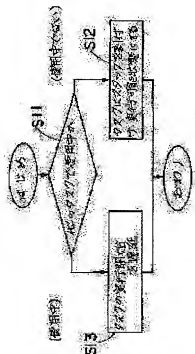
Priority number : 03 39269 Priority date : 08.02.1991 Priority country : JP

(54) STACK CONTROL SYSTEM AND MICROCOMPUTER

(57)Abstract:

PURPOSE: To provide a stack control system in a multitask processing system capable of easily performing processing to allocate a stack area to a task and improving memory working efficiency for the stack area.

CONSTITUTION: The same stack area is shared with plural tasks, and the allocation of the task to a shared stack area is controlled exclusively setting the execution of the task from start to completion as unit for individual shared task. For example, when the execution of the task is requested, it is decided whether or not another task which shares the stack area is using the shared task area (S11), and the task can be executed when another task is not being executed (S12), and transition to a state where the task can be executed is delayed when another task is being executed (S13).



LEGAL STATUS

[Date of request for examination]

11.12.1998

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

3212656

[Date of registration]

19.07.2001

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

特開平6-168145

(43) 公開日 平成6年(1994)6月14日

(51) Int.Cl. ⁵	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/46	3 4 0 F	8120-5B		
12/00	5 7 2	9366-5B		

審査請求 未請求 請求項の数9 (全 21 頁)

(21) 出願番号	特願平3-352297	(71) 出願人	000005108 株式会社日立製作所 東京都千代田区神田駿河台四丁目6番地
(22) 出願日	平成3年(1991)12月13日	(72) 発明者	清水 剛 東京都小平市上水本町5丁目20番1号 株式会社日立製作所武蔵工場内
(31) 優先権主張番号	特願平3-39269	(72) 発明者	竹山 寛 東京都小平市上水本町5丁目20番1号 株式会社日立製作所武蔵工場内
(32) 優先日	平3(1991)2月8日	(72) 発明者	小早川 学 東京都小平市上水本町5丁目20番1号 株式会社日立製作所武蔵工場内
(33) 優先権主張国	日本 (J P)	(74) 代理人	弁理士 玉村 静世
特許法第30条第1項適用申請有り 1990年12月18日~12月19日 社団法人トロン協会主催の「第7回トロンプロジェクト国際シンポジウム」において文書をもって発表			

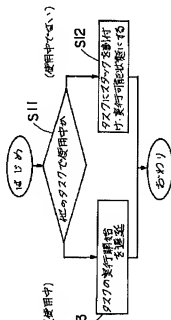
(54) 【発明の名称】 スタック制御方式及びマイクロコンピュータ

(57) 【要約】

【 図 1 】

【目的】 本発明の目的は、スタック領域をタスクに割り付ける処理が簡単であって、且つ、スタック領域のためのメモリ利用効率を向上させることができるマルチタスク処理システムにおけるスタック制御方式を提供することにある。

【構成】 複数のタスクに同一スタック領域を共有させ、個々の共有スタック領域に対しては、タスクの実行開始から終了までを単位として、前記共有スタック領域に対するタスクの割り付けを排他的に制御するものである。例えば、タスクの実行が要求されたとき、スタック領域を共有する他のタスクが当該共有スタック領域を使用しているかを判定し (S11)、他のタスクが実行中でなければ当該タスクを実行可能にし (S12)、他のタスクが実行中であれば当該タスクの実行可能な状態への遷移を遅延させる (S13) ようにする。



【特許請求の範囲】

【請求項1】 オペレーティングシステムの管理の下で複数のタスクを実行制御するマルチタスク処理システムにおいて、複数のタスクに同一スタック領域を共有させ、それらタスクの実行開始から終了までを単位として、前記共有スタック領域に対するタスクの割り付けを排他的に制御することを特徴とするスタック制御方式。

【請求項2】 前記排他的な制御は、タスクの実行が要求されたとき、スタック領域を共有する他のタスクが実行中であるかを判定する処理と、この判定結果により、実行中でなければ当該タスクを実行可能にする処理と、他のタスクが実行中であれば当該タスクの実行開始を遅延させる処理とを含むことを特徴とする請求項1記載のスタック制御方式。

【請求項3】 実行が要求されたタスクが割り付けられるべき共有スタック領域を、タスクに固有のスタック指示情報から求める処理を更に含むことを特徴とする請求項2記載のスタック制御方式。

【請求項4】 他のタスクが実行中であるかを判定する前記処理は、共有スタック領域に現在割り付けられているタスクが存在していることを指示するためのスタック管理変数を参照して行うことを特徴とする請求項2又は3記載のスタック制御方式。

【請求項5】 前記遅延させる処理は、共有スタック領域固有のスタック行列管理ポインタに実行開始が遅延されるタスクを繋ぐ処理であることを特徴とする請求項2乃至4の何れか1項記載のスタック制御方式。

【請求項6】 前記排他的な制御は更に、タスクの実行を終了したときに、そのタスクとスタック領域を共有して実行開始が遅延されている他のタスクが存在するか否かを判定する処理と、これにより存在すると判定されたときに、実行開始が遅延されている単数若しくは複数のタスクの中から選ばれた一つのタスクを実行可能にさせる処理とを含むことを特徴とする請求項2記載のスタック制御方式。

【請求項7】 実行開始が遅延されている他のタスクが存在するか否かを判定する前記処理は、スタック行列管理ポインタの値を参照して、当該ポインタの値が他のタスクに繋がれていないことによって判定することを特徴とする請求項6記載のスタック制御方式。

【請求項8】 選ばれた一つのタスクを実行可能にさせる前記処理は、当該タスクを、実行可能なタスクの待ち行列を形成するための実行待ち行列管理ポインタに繋ぐ処理であることを特徴とする請求項6又は7記載のスタック制御方式。

【請求項9】 マルチタスク処理されるべき複数のタスクがスタック領域を共有したとき、タスクにそのスタック領域を、タスクの実行開始から終了までを単位として排他的に割り付け制御するための、スタック制御領域を確保する機能を持つオペレーティングシステムを搭載し

てなる1チップ型のマイクロコンピュータ。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、オペレーティングシステムもしくはオペレーティングシステムプログラム（以下OSとも記す）のメモリ管理方式に関し、特にOSの管理の下で中央処理装置が複数のタスク（ユーザプログラム）を実行制御するマルチタスク処理システムにおいて、作業用のスタック領域を各タスクに割り当てるためのスタック制御方式、並びに同制御方式を採用可能なシングルチップマイクロコンピュータに関する。

【0002】

【従来の技術】 マルチタスク処理においては、システムの処理を独立して並列に処理可能な単位即ちタスクに分割して、これをオペレーティングシステム（OS）が管理しながらその実行を制御する。斯るマルチタスク制御OSは、タスクからのシステムコール（タスクの実行要求）に対してマイクロプロセッサやその他の資源を優先度などに応じて当該タスクに割り当ててそのタスクの実行を制御する。OSは、このときタスクにスタック領域を割り当てる。このスタック領域は、サブルーチンへのパラメータの引き渡しやサブルーチンからの戻り番地の保持並びにタスクの実行を中断したときのレジスタ情報の持帰・復帰処理などに利用される。タスクに割り当てられるスタック領域が各タスクに夫々固有の領域と数に、登録されるタスクの数又は登録可能なタスクの数に比例してメモリの使用量が増大する。特に仮想記憶をサポートしていないデータ処理システムにおいてはスタック領域に使用されるメモリ容量の増大が顕著である。そこでスタック領域のためのメモリ使用量を削減するための技術として特開昭63-86035号がある。この特開昭63-86035号は、スタック領域を一括プール化し、必要に応じてそのプールから動的にスタック領域を切り出してタスクに割り付ける方式を開示している。

【0003】

【発明が解決しようとする課題】 しかしながら、タスクの実行要求に応じてプール領域から動的にスタック領域を切り出してタスクに割り当てる方式においては、その動的な制御故にOSの処理が複雑化することがわかった。さらに動的にスタック領域を切り出し制御したとしてもスタック領域が不足しないようにそのプール領域の記憶容量を定める必要がある。すなわち、同時に実行されるタスクの数を予め計算した上で、プール領域の容量が決定される。したがって、スタック領域のためのメモリ利用効率が低下してしまうことがある。

【0004】 特に仮想記憶を有しないシステムにおいては、タスクに割り付け可能なメモリ容量には物理的な制限があるので、スタック領域のサイズ（記憶容量）をできるだけ小さなサイズにすることが必要になる。特に比

較的規模の小さなデータ処理システム、たとえば、中央処理装置と周辺の入出力（I/O）機能やランダム・アクセス・メモリ（RAM）、リード・オンリ・メモリ（ROM）などを1チップに搭載したシングルチップマイクロコンピュータによって制御部が構成されるようなデータ処理システムの場合には、限られた記憶容量しか持たない内蔵RAMを用いて複数のタスクを実行しなければならず、スタック領域のサイズを更に小さくする必要があることを本発明者は見出した。また、物理的に限られたメモリ領域の中で、タスクの実行要求毎に動的にスタック領域を切り出したり、あるいは夫々のタスクに固有のスタック領域を割り付ける方法は、実行可能な若しくは登録可能なタスク数に自ずから制限を及ぼすことにもなる。

【0005】本発明の目的は、スタック領域をタスクに割り付ける処理が簡単であって、且つ、スタック領域のためのメモリ利用率を向上させることができるマルチタスク処理システムにおけるスタック制御方式を提供することにある。

【0006】本発明の別の目的は、スタック領域として割当可能なメモリ領域が限られたデータ処理システムにおいて、実行可能な若しくは登録可能なタスク数を比較的多くすることができるマルチタスク処理システムにおけるスタック制御方式を提供することにある。

【0007】更に本発明はそのようなスタック制御方式を採用可能な1チップ型のマイクロコンピュータを提供することを目的とする。

【0008】本発明の前記並びにその他の目的と新規な特徴は本明細書の記述及び添付図面から明らかなるであろう。

【0009】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【0010】本発明に関するオペレーティングシステムには、同一の優先度を有する複数のタスクに一つのスタック領域を割り当てる様なスタック管理機能が持たせられる。したがって、一つのスタック領域は同一優先度を持つ複数のタスクに対して共用される。但し、一つのスタック領域を共通利用する複数のタスクは、それらが同時に実行されないタスクとされ、かつ、それらの実行順序が予め決定可能なタスクとされる。すなわち、タスクA、B及びCに一つのスタック領域を共用させる場合、各タスクA、B及びCの実行順序は、タスクAの実行終了後、タスクBが実行され、タスクBの実行終了後、タスクCが実行されるような実行順序とされるのが良い。言い換えるならば、共用スタック領域は、排他的に実行中の一つのタスク（A、B又はC）に割り付けられる。

【0011】また、本発明に関するOSは、一つのタスクに対して一つのスタック領域を専用として割り当てる

ことができる。たとえば、10個のタスクをOSが管理するとき、OSは、二つのタスクにそれぞれ専用のスタック領域を割り当て、ある同一の優先度を有する四つのタスクに一つの共用のスタック領域を割り当て、他の同一の優先度を有する四つのタスクに一つの共用のスタック領域を割り当てる。結果的に、全スタック領域の数は10個のタスクがあるにもかかわらず四つとされ得る。

【0012】

【作用】したがって、本発明に従えば、タスクの総数がn個（nは正の整数）であってもスタック領域の総数を $m < n$ （mは正の整数）とすることができ、スタック領域が割り当てられるべきランダム・アクセス・メモリの全メモリ容量に対する全スタック領域の割合を低減することができる。

【0013】その結果、シングルチップマイクロコンピュータの様に内蔵ランダム・アクセス・メモリ（RAM）の記憶容量が制限される場合であっても、スタック領域として利用される記憶容量は、上記RAMの全記憶容量のわずかとされ得るので、シングルチップコンピュータにマルチタスクのオペレーティングシステムプログラム（OS）を搭載することが可能となる。

【0014】

【実施例】まず最初に本発明に従うオペレーティングシステムに関し説明する。

【0015】図5に示されるように、マルチタスク制御用のオペレーティングシステム（OS）は複数のタスクP1乃至Piを並列的に管理してその実行を制御する。

【0016】図6には、OSがタスクを管理するとき同タスクが採り得るタスク状態遷移図の一例が示される。同図に示される遷移状態は、休止状態、実行可能状態、実行状態、及び待ち状態である。

【0017】休止状態とは、タスクがまだ起動されていない状態即ちタスクの実行要求がまだされていない状態、或いはタスクの実行が終了された状態である。実行可能状態とは、タスクを実行可能な状態であるが、優先度の高い他のタスクなどが実行されているためプロセッサが割り付けられず待っている状態である。前記実行状態とは、タスクにプロセッサが割り付けられて現在実行中の状態である。前記待ち状態とは、タスクが、プロセッサ以外の資源の獲得を待っている、又は何らかの事象の発生を待っている状態である。

【0018】本発明に従うOSは、複数のタスクに同一スタック領域を共有させるようなメモリ管理機能を含ませられている。本発明のOSは、例えば図4示されるように、優先度が同一レベルであって実行順序が相互に予め決定されるようなタスク群に同一のスタック領域を共有させるように、スタック領域の割り当てを制御する。図4では、タスクP2、P3、P4のスタック領域B、タスクP5、P6、P7のスタック領域C、タスクP8、P9、P10のスタック領域Dが夫々共有スタック

5

領域とされる。

【0019】尚、タスクP1, P11には、それぞれスタック領域A及びEが専用スタック領域として割り当てられる。そして、本発明のOSは、個々の共有スタック領域に対しては、タスクの実行開始から終了までを単位として、前記共有スタック領域に対するタスクの割り付けを排他的に制御する。

【0020】図1は、本発明のOS内に含ませられる前記排他的な制御のプログラムフローチャートの例が示されている。すなわち、タスクの実行が要求されたとき、スタック領域を共有する他のタスクが実行中であるか否か（換言すれば当該共有スタック領域を他のタスクが使用しているか否か）が判定される（S11）。その結果他のタスクが実行中ではないと判定されれば、OSは当該タスクを実行可能にする（S12）。一方、他のタスクが実行中であると判定されれば、OSは当該タスクの実行開始即ち実行可能な状態への遷移を遅延させる（S13）。

【0021】図3は、タスクの実行開始を遅延させる前記処理の一例を示している。たとえば、図4のようにスタック領域Bを利用してタスクP2が実行状態にあるときに、同スタック領域を共有する他のタスクP3, P4の実行が要求されると、当該共有スタック領域Bに固有のスタック行列管理ポインタ（後述するBp）にそれらタスクP3, P4を繋いで待ち行列を形成するような処理を採用することができる。この待ち行列は、例えば、先入れ後出し（FIFO）のキューイングテーブルで構成される。

【0022】実行要求されたタスクが割り付けられるべきスタック領域を容易に認識可能にする為、各タスクに対し、それが割り付けられるべきスタック領域を指示するスタック指示情報若しくはスタック指示データ（後述するSPD）が設けられる。他のタスクが実行中であるか否かを判定できるようにする為、共有スタック領域に現在割り付けられているタスクが存在しているか否かを指示するためのスタック管理変数（Bd）が共有スタック領域Bに対して設けられる。したがって、OSはタスクが実行要求されると、そのタスクが割り付けられるスタック領域を認識するために、そのタスクのスタック指示情報をアクセスする。その結果、スタック指示情報が、例えば、共用スタック領域を指示している時、上記共有スタック領域の為に設けられたスタック管理変数をアクセスする。その結果、上記スタック領域に現在割り付けられているタスクが存在すると判定されれば、上記実行要求されているタスクは、待ち行列に入れられる。このことは、後述される図12〜図14で詳細に説明する。

【0023】共有スタック領域に対するタスクの前記排他的な割り付け制御の一貫として、実行開始が遅延されたタスクの取扱いを考慮する場合には、例えば図2に示

6

されるように、一つのタスクの実行を終了したとき、スタック領域を共有すると共に実行開始が遅延されているところの他のタスクが存在するか否かを判定する（S21）。この判定の結果、実行開始が遅延されている他のタスクが存在する場合には、当該実行開始が遅延されている単数若しくは複数のタスクの中から選ばれた一つのタスクが実行可能状態にされる（S22）。

【0024】実行開始が遅延されている他のタスクが存在するか否かを容易に判定する為、図5に示されたスタック行列管理ポインタBpの値を参照して、当該ポインタの値が他のタスクを指しているか否かを判定する。

【0025】当該実行開始が遅延されている複数のタスクの何れを選択するかはOSによる制御は、その待ち行列に含まれるタスクの優先度の高いものから、或いは待ち時間の長いものから、又はタスクの実行に要する時間が短いものから選択させたり、システムの要求に応じて適宜選択させることもできる。選ばれた一つのタスクを実行可能にさせる前記処理を比較的簡単にする為、実行可能なタスクを繋ぐためのレーディータスク行列管理ポインタを設け、これにそのタスクを繋ぐようにされる。

【0026】上述のスタック制御方式は、1チップ型のマイクロコンピュータのオペレーティングシステムとして有効である。その際マルチタスク処理されるべき複数のタスクにスタック領域を共有させるとき、一つのタスクの実行開始から終了までを単位としてそのタスクにそのスタック領域を排他的に割り付け制御するための共有スタック制御領域を確保する機能、すなわちスタック行列管理ポインタBp、スタック指示情報SPD、スタック管理変数Bdを設け、それら管理する機能をオペレーティングシステムにもたせ、該オペレーティングシステムを搭載すればよい。

【0027】この様にオペレーティングシステムを構成すれば下記の効果を得ることができる。複数のタスクに同一スタック領域を共有させ、タスクの実行開始から終了までを1単位として共有スタック領域に対するタスクの割り付けを排他的に待ち行列方式で制御するので、一括メモリプール化によるスタック領域の動的割り付け制御や仮想記憶を利用する制御に比べて、スタック領域をタスクに割り付ける処理プログラムを簡単にできる。同時に、共有スタック領域を設定するので、マルチタスク処理システムにおけるスタック領域のためのメモリ容量を低減可能であり、また、共有スタック領域を複数タスクで利用するので、メモリ利用効率を向上させることができる。このメモリ利用効率は、スタック領域に対する共有の度合いを増す程に向上する。

【0028】マルチタスク処理システムにおけるスタック領域のためのメモリ利用効率の向上は、スタック領域として割り当てるメモリ領域が限られたデータ処理システムにおいても、実行可能な若しくは登録可能なタスク数をさほど制限しないように働く。

【0029】さらに、1チップ型のマイクロコンピュータ（シングルチップマイクロコンピュータ）のようにスタック領域として割当可能なメモリ領域が限られた中でも、管理すべきタスクを構成する各種ユーザプログラム若しくはアプリケーションプログラムに対してOSの柔軟な対応を可能にする。

【0030】以下、1チップ型のマイクロコンピュータ（シングルチップマイクロコンピュータ）を用いたデータ処理システムに本発明を適用した一実施例について説明する。

【0031】図7に示されるマイクロコンピュータ10は、特に制限されないが、オペレーティングシステムプログラム（OS）やタスク（ユーザプログラム若しくはアプリケーションプログラム）といった動作プログラムを実行する中央処理装置（CPU）11、上記OSやタスクといった動作プログラムなどを保有するROM（リード・オンリ・メモリ）12、前記中央処理装置11の作業領域やデータの一時記憶領域、例えばスタック領域などとして利用されるRAM（ランダム・アクセス・メモリ）13、入出力（I/O）インタフェース14、タイマ15を含み、それらはアドレスバス16、データバス17、及びコントロールバス18によって結合されている。上記マイクロコンピュータ10は、公知の半導体集積回路製造技術によって単結晶シリコンのような1個の半導体基板上に形成されている。

【0032】上記タイマ15は、端子T1～T3を介して、外部に設けられたモータM1～M3の発生するパルス信号S1～S3を受けて、パルス信号の数をカウントする。前記I/Oインタフェース14は、キーボード51からの入力データを端子T4～T10を介して内部データバス17に供給し、内部データバス17上のデータを端子T14～Tnを介して表示装置（ディスプレイ）52に出力する。

【0033】図7のデータ処理システムの動作については後で詳述する。

【0034】上記OSは、タスク管理、タスク付属同期管理、同期・通信管理、時間管理、割り込み管理などの機能を含む。以下に、その概要を示すと、（1）タスク管理（スケジューラを含む）、CPU（Central Processing Unit）11をタスクに割り付ける順序やタスクの起動・終了など、タスクの状態を管理し、このときタスクは優先順位の高いものから順にCPUに割り付けられ、（2）タスク付属同期管理は、タスクの実行中断・再開など、タスクの基本的な同期処理を行い、（3）同期・通信管理は、イベントフラグ、セマフォ、メールボックスの三つの機能があり、タスク間の同期・通信処理を行い、（4）時間管理は、時間の管理を行うと共に、タスクの実行制御のための時間監視を行い、（5）割り込み管理は、割り込み発生時に割り込み処理ハンドラを起動し、割り込み処理やタスクへの割

込み発生の連絡を行う。

【0035】尚、本発明は、OSのタスク管理スタック領域割付け機能に関するものなので、上記（2）～（5）に示される管理内容に関しては詳細に説明されないが、当業者にとって容易に理解されるであろう。

【0036】前記中央処理装置11は、ROM12に記憶されたOSを核として各種タスクを所定の手順に従って実行制御する。CPU11は、実行すべきタスクの実行に際し、そのタスクを構成する動作プログラムをROM12から読み込んで解読するため、アドレスバス16に上記タスクの先頭アドレスに対応するアドレス信号を送出する。上記ROM12はアドレス信号に対応して、プログラムデータをデータバス17に送出する。CPU11は、上記プログラムデータを読み込んでこれを解読し、その解読結果に応じて各種演算制御や周辺回路の制御を行う。したがって、CPU11は、OSを実行している動作モード（スーパーバイザモード）とタスク（ユーザプログラム）を実行している動作モード（ユーザモード）とを有している。上記OSを実行しているCPU11が、以下において説明される各種のタスク制御を実行していることと見なされる。尚、図7には中央処理装置11が保有する各種レジスタとして、たとえば、16ビットのプログラムカウンタPCや8ビットのコンディショナルコードレジスタCCRのような制御レジスタ、データレジスタやアドレスレジスタなどとして使われる汎用レジスタ群Rn、及び現在実行中のタスクに割り付けられているスタック領域のトップアドレス（先頭アドレス）を指す16ビットのスタックポインタSTKPが代表的に図示されている。

【0037】汎用レジスタ群Rnは、たとえば、それぞれが16ビットとされる7本のレジスタR0～R6を含む。7本のレジスタR0～R6の内、たとえば、4本のレジスタR0～R3は、各タスクの文脈（プログラム）内でワークレジスタとして定義される。この様にする理由は、スタック領域として準備する記憶領域の容量を低減する為である。しかし、全ての汎用レジスタR0～R6をプログラム内でワークレジスタとして定義しても良い。このスタックされるべき汎用レジスタRnの本数は、システム構築時に特定することができる。

【0038】図8は、マイクロコンピュータ10のシングルチップモードでのアドレスマップを示している。ROM12は16kバイトとされ、16進アドレスH'Fが付けられる。表示するとH'0000からH'3FFFをしめる。ROMアドレスH'0000～H'003Dは、本発明と直接関係しないので詳細には説明されないが、インターラプトベクターテーブルを含む。H'03E～H'3FFFは、図9に示される様に、RAM13のアドレス領域を後述される図10の様に初期設定するためのデータテーブルを記憶する領域99、OSプログラムの記憶された領域100、タスクP1～P11

のプログラムが記憶された領域101~111、及びスタック指示データ(SPD)テーブルの記憶された領域112を含む。このSPDテーブルは、各タスクP1~P11に対応して設けられるスタック指示データSPD1~SPD11を含む。これら各スタック指示データSPD1~SPD11は、各タスクが利用するスタック領域を示すための情報が記憶される。たとえば、各スタック指示データSPD1~SPD11は、対応するスタック領域のボトムアドレス、すなわち、アドレスマップ上で割り当てられる各スタック領域の最上位アドレス(終端アドレス)とされる。尚、ここで図8のアドレスマップ上において、アドレスの上位側とは、アドレスの大きな方向(H'FFFF方向)を示し、アドレスの下位側とはアドレスの小さな方向(H'0000方向)を示すと見なされる。さらに、スタック領域にあるデータがスタックされる場合、スタックポインタSTKPの値は、ボトムアドレス(スタック領域のアドレスマップ上の最上位アドレス)からスタックされるべきデータのバイト数を引いた値へと更新されるものとする。

【0039】図4の様に、各スタック領域を各タスクに設定する場合、各スタック指示データSPD1~SPD11には夫々以下の様なデータがセットされる。

SPD1	スタック領域Aのボトムアドレス
SPD2~4	スタック領域Bのボトムアドレス
SPD5~7	スタック領域Cのボトムアドレス
SPD8~10	スタック領域Dのボトムアドレス
SPD11	スタック領域Eのボトムアドレス

【0040】したがって、CPU11は、上記スタック指示データテーブル112内の所望のスタック指示データ記憶領域のアドレスをアドレスバス16に出力し、リード状態を示すコントロール信号をコントロールバス18に出力することによって、上記所望スタック指示データ記憶領域にアクセスする。そして、CPU11は、それに応答してROM13からデータバス17に出力されたアドレスデータをリードすることによって、タスクがどのスタックに割り当てられているかを識別する。

【0041】RAM13は、図8に示される様に、512バイトとされ、16進アドレスで表示すると、H'FD80~H'FFFFFに割り当てられる。RAMアドレスH'FD80~H'FFFFFに割当てられる。RAMアドレスH'FD80~H'FFFFFは、図10に示される様に、16バイトのOSスタック領域201、それぞれ12バイトとされる各タスクのタスク管理テーブルTCB1ないしTCB11を記憶する領域202~213、それぞれ20バイトとされるスタック領域AないしE、それぞれ1バイトとされるスタック管理変数AdないしEdを記憶する領域214ないし218、それぞれ2バイトとされるスタック行列管理ポインタApないしEpを記憶する領域219ないし223、実行待ち行列管理ポインタXpを記憶する領域24、及び247バ

イトのCPU11の作業領域224を含む。

【0042】尚、スタック管理変数AdないしEd及びスタック行列管理ポインタApないしEpの領域214ないし223は共有スタック制御領域とされる。

【0043】スタック管理変数領域214~218には対応するスタック領域が使用されているか否かを示す情報、又は対応スタック領域(A~E)を使用しているタスクを特定するための情報、例えば、そのタスクの番号情報がCPU11によって記述される。

【0044】スタック行列管理ポインタApないしEpは対応する共有スタック領域の空きを持っているタスクを待ち行列に繋ぐためのポインタとされる。

【0045】図10からも明かなようにスタック領域(A)とこれに対応する共有スタック制御領域214、219は連続するアドレスにマッピングされている。従って、CPU11は、例えば、タスクP2の実行要求が他のタスク又はキーボード51からされたとき、同タスクP2の実行に利用するスタック領域がBであることを、上記内部アドレスバス16にスタック指示データSPD2を示すアドレスを出力し、内部データバス17に出力されるスタック指示データSPD2の内容から認識することができる。その共有スタック領域Bを他のタスクが使用しているか否かは、スタック指示データSPD2の内容が指しているアドレスの上位側アドレスをCPU11が参照(アクセス)すれば、そこに割り当てられているスタック管理変数領域215の保持情報Bpから認識できる。さらに当該スタック領域Bを他のタスクが使用している場合に当該タスクをスタック待ち行列に繋げたり、あるいはスタック待ち行列に繋がつているタスクを認識したりする場合には、CPU11が、さらにその次アドレスのスタック行列管理ポインタBpの値を参照して行列を辿ったりすればよい。したがって、それらの手順は極めて簡単になる。1個のタスクに専用化されたスタック領域A、Eのためのスタック管理変数Ad、Edとスタック行列管理ポインタAp、Epは当該スタック領域が1個のタスクに占有されていることと同じと見て同様に認識させるために利用するものである。したがって、専用スタック領域であることを示す特別なフラグなどを設け、それをCPU11が参照するような手順(プログラム)を採用すれば、専用スタック領域A、Eに関してはスタック管理変数領域214、218やスタック行列管理ポインタ領域219、223をRAMに割り当てなくてもよい。この様にすることによって、RAM13のメモリ容量を有効に利用できる。

【0046】図8のアドレスマップに示される様に、12バイトの内蔵レジスタは、16進アドレスで表示すると、H'FF90からH'FFFFFに割り当てられる。これらのレジスタは、I/Oインタフェース14内のレジスタ及び内蔵周辺回路であるタイマ15内のレジスタに対応している。これらのレジスタに関しては、本

発明と直接関係しないので、詳細に説明されないが、当業者にとって容易に理解されるであろう。

【0047】図11にはタスク管理テーブルTCBの一例が示される。タスク管理テーブルTCBは、対応タスクの開始アドレス指定領域20、タスクを待ち行列に繋ぐための各種ポインタ群21、図6に示されるようなタスクの状態が記述される領域22、及びタスクの優先度を記述する領域23などを含む。例えば実行可能状態のタスクを繋げる待ち行列の先頭はCPU11によって実行待ち行列管理ポインタ24に記述される。その記述内容は対応タスクにおけるタスク管理テーブルのポインタ21Aのアドレスである。夫々のタスク管理テーブルにおけるポインタを利用して、CPU11は次々にタスクを繋げる様々ことができる。同様にCPU11による前記タスク待ち行列管理ポインタへの記述は、対応タスクにおけるタスク管理テーブルのポインタ21Bのアドレスである。実際には一つのタスクがスタック待ち行列と実行待ち行列の双方に繋がれることはないので前記ポインタ21Aと21Bは同一の領域が使用される。

【0048】図12にはタスク、スタック領域、及びタスク制御領域の関係が機能的に示されている。同図には共有スタック領域Bと専用スタック領域Aが代表的に示されている。タスクP2、P3、P4の夫々が利用するところのスタック領域がBであることは、夫々のスタック指示データSPD2、SPD3、SPD4が指しているRAM13のアドレスによって認識される。タスクP1が利用するスタック領域がAであることは該タスクに対応するスタック指示データSPD1が指しているRAM13のアドレスによって認識される。同図においてタスクP2がスタック領域Bに割り付けられて実行状態にあるとき、スタック管理変数領域Bdには当該スタック領域Bを利用しているタスクP2を示す情報(タスク番号)がCPU11によって書き込まれている。このとき、スタック領域Bを共有する他のタスクP3の実行要求があると、CPU11はスタック行列管理ポインタBpにタスクP3に対応するタスク管理テーブルTCB3のポインタアドレスをスタック行列管理ポインタBpに記述して繋ぐ。続いてタスクP4の実行要求があると、CPU11は更にタスク管理テーブルTCB3のポインタにタスクP4に対応するタスク管理テーブルTCB4のポインタアドレスを記述してスタック待ち行列を形成する。一方タスクP1の実行が要求されていない状態においては、それ専用のスタック領域Aに対応するスタック管理変数領域Adは「空き」(即ち未使用)を意味する情報がCPU11によって記述されており、また、スタック行列管理ポインタApにもタスクを繋ぐための情報が記述されず「空き」の状態にされている。

【0049】図13にはタスクの実行要求に際してのスタック領域に対するタスク制御フローチャートが示されている。この内容は、図1に対応された更に詳細

なものに対応する。

【0050】まず、休止状態のタスクの中から一つのタスクに対して実行要求があると、CPU11は要求されたタスクのスタック指示データ(SPD)から当該タスクが利用するスタック領域を求める(S31)。例えば、図12においてタスクP2の実行要求があると、CPU11はタスク2に対応するスタック指示データSPD2の指すRAM13のアドレスによってスタック領域Bを認識する。

【0051】次に、CPU11はそのスタック領域Bを使って実行状態にある他のタスクが存在するか否かを判定するための情報を、スタック領域Bに対応するスタック管理変数領域Bdから求める(S32)。CPU11は該スタック管理変数領域の参照のため、前記スタック指示データを利用する。例えば、図12のタスクP2に着目すると、CPU11はスタック指示データSPD2が指すRAM13のアドレスの上位側アドレスをアクセスすることによって、対応するスタック管理変数領域Bdを参照する。

【0052】前記スタック管理変数領域を参照した結果、その領域が「空き」の状態であれば、CPU11は当該スタック領域は未使用であると判定する。一方、当該領域にタスクの番号などの情報が保持されているとき、CPU11は他のタスクによって使用中であると判定する(S33)。例えば、図12に示されるように、スタック管理変数領域BdにタスクP2を意味する情報が記述されている状態において、そのスタック領域Bを共有するタスクP3の実行要求に従って、CPU11がスタック管理変数領域Bdを参照した場合には、タスクP2が当該スタック領域Bに割り付けられていると判断する。また、図12においてスタック管理変数領域Adが「空き」の状態において、そのスタック領域Aを利用しようとするタスクP1の実行要求に従って、CPU11がスタック管理変数領域Adを参照した場合には、当該スタック領域Aは未使用であると判定する。

【0053】ステップS33により使用中であると判定された場合には、今回実行要求されたタスクをスタック行列管理ポインタに繋ぐ(S34)。そして斯るタスクを待ち状態にして実行を遅延させる(ステップS35)。例えば、図12に示されるように、タスクP2が実行状態にあるとき、そのスタック領域Bを共有する他のタスクP3の実行要求があった場合、CPU11は、そのタスクP3に対応するタスク管理テーブルTCB3のポインタ(図11のポインタ21B)アドレスをスタック行列管理ポインタBpに記述して繋ぐ。続いてタスクP4の実行要求があると、更にCPU11はタスク管理テーブルTCB3のポインタにタスクP4に対応するタスク管理テーブルTCB4のポインタアドレスを記述してスタック待ち行列を形成する。

【0054】ステップS33により未使用であると判定

された場合、CPU 11は対応するスタック管理変数領域に該タスク番号を記述する(S36)。そして当該タスクが実行可能状態にされる(S37)。

【0055】特に、本実施例に従えば、スタック領域を共有するタスクとスタック領域を占有するタスクとが混在され、スタック領域AやDのような専用スタック領域に対してもスタック管理変数が割り当てられている。そのため、タスクの実行要求に際してのスタック領域割り付け制御手順は、共有スタック領域に対してもまた専用スタック領域に対しても同じとすることができる。したがって、OSのプログラム作成が簡単化され得る。

【0056】図14にはタスクの実行終了に際してのタスク割り制御の一例フローチャートが示される。この内容は、図2に対応し、実行開始が遅延されたタスクの取扱いとスタック領域の解放を考慮した更に詳細な内容を有する。

【0057】先ず、タスクの実行が終了されると、CPU 11は、そのタスクが占有していたスタック領域をスタック指示データから求める(S41)。次に、CPU 11はそのスタック指示データが指すRAM 13のアドレスを元に対応するスタック行列管理ポインタを参照してスタック待ち行列にタスクが繋がれているかを判定するための情報を取得する(S42)。そして、CPU 11はその情報に基づいてスタック待ち行列を構成するタスク管理テーブルが存在するか否かを判定する(S43)。

【0058】ステップS43の判定結果により該当するタスク管理テーブルが存在しない場合には、CPU 11は対応するスタック管理変数に「空」を意味する情報を書き込む(S44)。これにより、当該スタック領域は、その後実行要求された所定のタスクに割り付け可能になる。

【0059】ステップS43の判定結果によりスタック待ち行列を構成するタスク管理テーブルが存在している場合には、CPU 11はスタック待ち行列に含まれる一つのタスク管理テーブルを当該スタック待ち行列からはずして当該スタック待ち行列を更新する(S45)。どのタスク管理テーブルをはずすかは図2で説明した通り、スタック待ち行列に含まれるタスクの優先度、或いは待ち時間の長さ、又はタスクの実行に要する時間の長短などを考慮して決定される。そして、CPU 11はスタック待ち行列からはずしたタスク管理テーブルに対応するタスクの番号などを対応するスタック管理変数領域に記述する(S46)。さらに、CPU 11はスタック待ち行列からはずしたタスク管理テーブルを今度は実行待ち行列に加えて当該タスク管理テーブルに対応するタスクを実行可能な状態にする(S47)。

【0060】上記スタック領域は、特に制限されないが、タスク内におけるサブルーチンへのパラメータ引き渡しやサブルーチンからの戻り番地の格納領域、さらに

は相対的に優先度の高いタスクの実行要求などによってそのときCPUが割り付けられて実行状態にあったタスクが中断されるようなときに当該中断されるべきタスクの実行再開に必要な各種レジスタ情報の待避領域などとして利用される。このレジスタ情報の待避領域には、図7に示されるようなプログラムカウンタPCの値、コンディションコードレジスタCCRの保有値、スタックポインタSTKPの値、並びに汎用レジスタ群Rnの保有データなどが待避される。

【0061】図15にはタスクの実行状態におけるスタック領域の利用態様の一例が示される。例えばタスクP2の実行状態に着目すると、その過程において「KEISAN」というサブルーチンに分岐するとき、当該タスクP2が割り付けられる共有スタック領域Bにはそのサブルーチンのためのパラメータ「X、Y」が引き渡されると共に、サブルーチンからの戻り番地例えば「1000番地」が保持される。このときスタックポインタSTKPはスタック指示データSPD 2の保有アドレスによってプリセットされ、これをスタック領域Bの開始番地として保有しており、スタック動作毎に順次デクリメントされていく。そして、サブルーチン「KEISAN」の実行途上において、当該タスクP2よりも優先度の高いタスク(タスクP2とは異なるスタック領域を利用する)例えばタスクP1の実行要求によってそのサブルーチン「KEISAN」を中断するときは、それまでに利用していた各種レジスタの情報例えばプログラムカウンタPC、コンディションコードレジスタCCR、スタックポインタSTKPの値などをスタック領域Bに待避し、その後当該タスクP2の実行を中断する。

【0062】ここで、タスクの登録即ちタスクをOSに認識可能にする手法は、OSとアプリケーションプログラムとを合成してROMへのロードモジュールを構成するようなプログラム構築時にROM上のテーブルでOSに認識可能にするやり方と、タスクの生成や削除を行うシステムコールを用いる手法がある。このとき、上記説明では、どのタスクが何れのスタック領域を共有するかは、夫々のタスクと対を成すスタック指示データによってOSが認識する。したがって、上記共有スタック領域に対するタスクの割り付け制御をサポートするOSを利用したシステムにおいて、登録可能なタスク数が全スタック領域の容量に対して余裕があるようなアプリケーションプログラムを対象とする場合、必ずしも共有スタック領域を設定しなくてもよい場合があり、このようなとき全てのスタック領域をタスクに1対1対応する専用領域とすることもできる。全てのスタック領域が専用領域である場合には、スタック管理変数領域やスタック行列管理ポインタのようなスタック制御領域を確保しなくてもよく、タスクの起動要求に応じて対応スタック領域を単に当該スタック領域に割り付ければよい。これを考慮すると、上記共有スタック領域に対するタスクの割り

15

付け制御をサポートするOSには、タスクの登録時に複数のタスクによるスタック領域共有状態を認識したときにスタック管理変数領域やスタック行列管理ポインタのようなスタック制御領域を確保する機能を与えておくことにより、システムに最適なスタック領域割り付け制御を行うことができる。

【0063】図16は、図7のデータ処理システムに適用されるOSとタスクとのタイムチャートの例を示している。尚、同図には理解を容易とするため、タスクP1ないしP5のみが示される。尚、これらタスクP1ないしP5のスタック領域及び優先度は図4に就うものとする。

【0064】上記データ処理システムが起動されると、それに応答して初期化プログラム（図示されていない）、たとえば、タスクP11が実行され、RAM13の図9に示される初期化データテーブル9.9に従って、図10の様にRAM13のアドレスが初期化される。上記初期化プログラムの最後プログラム語には、例えばタスクP1を実行要求するための命令str_tskP1が入れられており、上記OSは、その命令に基づいて、タスクP1を実行可能状態とする。すなわち、タスクP1内の命令str_tskはシステムコール命令すなわち、OS内のタスクを実行状態とするためのサブルーチンタスク起動（start-task）呼びだす命令とされる。尚、図13に示されるプログラムフローチャートがこのサブルーチンに対応する。

【0065】タスクP1は上記データ処理システムのメインプログラムと見なされる。モータM1～M3を所定の速度で回転させる様なプログラム及び速度調整のためのプログラムを含む。図示される様に、タスクP1の内部プログラムには、タスクP2、P3、P4を実行要求する命令str_tskP2、str_tskP3、str_tskP4が含まれる。したがって、タスクP1がタスクP2、P3及びP4を次々と実行要求する。その結果、システムコール命令str_tskP2に応答し、OS内のサブルーチンstart-taskが起動されタスクP2に共有スタック領域Bを割りつけて実行可能状態とする。そして次のシステムコール命令str_tskP3に応答して再度サブルーチンstart-taskが起動されタスクP3に共有スタック領域Bを割りつけて実行可能状態とする。上記タスクP3は上記タスクP2とスタック領域Bを共有しかつ、既にスタック領域BにタスクP2が割り当てられているので、タスクP3はスタック待ち行列につながれることになる。同様に、システムコール命令str_tskP4に応答して再度サブルーチンstart-task（start-task）が起動される。上記タスクP4もまたスタック待ち行列につながれることになる。したがってスタック待ち行列は、タスクP3とそれにつづくタスクP4とがつながれることになる。この時

16

同様に実行待ち行列にもタスクP3とタスクP4がつながれることになる。

【0066】その後、タスクP1内のシステムコール命令str_tskP1によって、タスクP1は自らを図6の待ち状態とする。すなわち命令stptskP1はOS内のサブルーチンスリープタスク（sleep-task）呼びだす命令とされる。上記サブルーチンスリープタスクは、それが起動されると、タスク管理テーブルTCBのタスクの状態を記憶する領域2.2のデータを待ち状態を示すデータとし、対応するタスクを待ち状態とする。

【0067】次に、実行可能状態とされているタスクP2が実行される。タスクP2はたとえば、モータM1、M2、M3から出力される各パルス信号S1～S3をタイム15で計測するプログラムとされる。

【0068】タスクP2のプログラム語の最後にはシステムコール命令ext_tskP2が書かれている。このシステムコール命令ext_tskP2はOSのサブルーチンタスク終了（exit-task）呼び出す命令とされ、それによって図14のプログラムフローチャートが実行される。したがって、システムコール命令ext_tskP2が実行されるとタスクP2は終了される（休止状態とされる）ことになり、スタック行列管理ポインタが調べられる。

【0069】この実施例の場合、タスクP1からタスクP2ないしP4の実行要求が出されているので、OSはタスクP2を待ち行列からはずし、タスクP3を実行可能状態とする。そして、タスクP3が実行される。タスクP3は、タスクP2で得られた計測データをCPU11で演算し、各モータM1ないしM3の回転速度データを求めるプログラムとされる。

【0070】次に上記タスクP3の実行はシステムコール命令ext_tskP3にしたがって終了される（休止状態にされる）と、OSは、タスクP3を待ち行列からはずし、タスクP4を実行可能状態とする。そして、タスクP4が実行される。

【0071】タスクP4は、上記タスクP3で得られた速度データを表示装置2に表示するプログラムとされる。タスクP4の実行は、システムコール命令ext_tskP4に従って、終了される（休止状態とされる）。タスクP4のプログラムにおいて、上記システムコール命令ext_tskP4の前には、システムコール命令wup_tskP1が記入されている。このシステムコール命令wup_tskはOS内のサブルーチンウェイクアップタスク（wake-up-task）呼びだす命令とされ、それによって待ち状態とされているタスクを実行可能状態とする。したがって、図16において、タスクP1が待ち状態とされているタスクであり、上記システムコール命令wup_tskP1によって、タスクP1が実行可能状態に復帰される。すなわ

ち、タスクP1のタスク管理テーブルTCB1内のタスクの状態を記憶する領域22のデータが、待ち状態を示すデータから実行状態を示すデータに書き変えられる。そして、タスクP1が実行される。これによってタスクP3の結合に基づいて、モータの回転速度を所定値に設定するタスクP1内のプログラムが実行されようとする。

【0072】図16では、その後キーボード51の入力によって、システムコール命令`istr_tskP5`が発生されている。このシステムコール命令`istr_tsk`は、外部キー入力などのデータにตอบสนองして発生される様な命令で、非タスク部からのタスク実行要求命令とされる。この命令`istr_tsk`は、OS内のサブルーチン非タスク部のタスク起動(`start task for task independent portion`)を呼び出す命令である。そのサブルーチンによって、タスクP1の実行は中断され、待ち状態とされ、タスクP5が実行される。したがって、タスクに依存しない外部入力によってもタスクの実行要求を発行することができる。前記タスクP5は、たとえば、モータM1～M3の回転を停止させる様な出力信号01～03をシングルチップマイコン10から出力させるプログラムとされる。この様にOSはタスクP1ないしP5の実行を制御する。

【0073】上記実施例によれば以下の作用効果を得るものである。

【0074】(1) 複数のタスクに同一タスク領域を共有させ、タスクの実行開始から終了までを1単位として共有タスク領域に対するタスクの割り付けを排他的に制御することにより、一括メモリプール化によるタスク領域の動的割り付け制御や仮想記憶を利用する制御に比べて、タスク領域をタスクに割り付ける処理を簡単にすることができる。これと同時に、マルチタスク処理システムにおけるタスク領域のためのメモリ利用効率を向上させることができる。このメモリ利用効率は、タスク領域に対する共有度合いを増す程向上する。タスク領域に対する共有度合いを増すには、一連の実行順序が一義的に決定される性質のタスクを多く採用すればよく、換言すれば、往々にして動作パターンが限られるような機器の制御に最適である。

【0075】(2) 上記により、マルチタスク処理システムにおけるタスク領域のためのメモリ利用効率が向上すれば、タスク領域として割当可能なメモリ領域が限られた中でも、実行可能な若しくは登録可能なタスク数が制限される事態を防止することができる。したがって、1チップ型のマイクロコンピュータのようにタスク領域として割当可能なメモリ領域が限られた中でも、タスクを構成する各種ユーザプログラム若しくはアプリケーションプログラムに対してOSを柔軟に対応させることができる。

【0076】(3) 共有タスク領域と専用タスク領域が混在される場合に、その専用タスク領域に対してもタスク管理変数領域やタスク行列管理ポインタのようなタスク制御領域を確保することにより、双方のタスク領域に対するタスクの割り付け制御に図13及び図14で説明したような同一手順を採用することができ、タスク割り付け制御を一層簡素化することができる。

【0077】(4) 従来のタスク領域一括プール化によるタスク領域動的割り付け制御に対し、タスク待ち行列によりどのタスクがどのタスクによって実行遅延されているか否かが明らかになり、これをその他適宜のプロセスシーケンスにも利用することが可能になる。

【0078】以上本発明者によってなされた発明を実施例に基づいて具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。

【0079】例えば上記実施例では、共有タスク領域に対するタスクの割り付け制御により実行を遅延させるべきタスクをポインタで繋いでタスク待ち行列を形成して管理する場合について説明したが、実行を遅延させるべきタスクの番号情報若しくはそのタスク管理テーブルアドレスなどを特定の領域に順番に書き込んで管理するような配列管理などの手法を採用することもできる。但し斯る手法は、タスク待ち行列に比べてメモリ使用量が僅かながら増えると予想される。

【0080】また、OSのリアルタイム性若しくはタスクの緊急的な実行を考慮する場合に、任意のタスクによって共通利用可能なタスク領域を予め確保しておき、非同期で発生する事象に対処しなければならないようなタスクの優先的な実行にも対応できるようにすればよい。

【0081】また、共有タスク領域に対するタスクの割り付け制御を実現するに当たり、どのタスクがどのタスク領域を共有するかをOSに認識可能にするための手法は上記実施例の説明に限定されず、明示的若しくは暗黙的にタスクとタスク領域とを論理的に結合する以下の手法がある。

(a) タスク領域を共有するタスク或いはタスク群をテーブルを用いてOSに認識させる。

(b) ロードモジュールの生成若しくはタスクの生成などのタスク登録時に、登録されるタスクが使用するタスク領域を指定させ、その際他のタスクと同じタスク領域が指定されているときは、その指定されたタスク領域を共有させる。

(c) 同一優先度を持つタスクには同一タスク領域を自動的に共有させる。

(d) タスク登録時に他のタスクを指定させ、指定されたタスクが使用するタスク領域を当該登録されるべきタスクにも共有させる。

(e) タスクを登録したタスクのためのスタック領域を当該登録されるべきタスクに共有させる。

【0082】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野である1チップ型のマイクロコンピュータに適用した場合について説明したが、本発明はそれに限定されず、種々のコンピュータシステムに広くて適用することができる。

【0083】本発明は、少なくともOSによってタスクを管理してその実行を制御するマルチタスク制御を条件とするものに広く適用することができる。

【0084】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【0085】すなわち、1個のスタック領域を複数のタスクが共有することにより、スタック領域のためのメモリ利用効率を向上させることができるという効果がある。

【0086】これにより、従来メモリ容量不足によって満足いくマルチタスク処理が行えなかったシステムにおいてもOSによるマルチタスク処理の実現が可能になるという効果がある。

【0087】それと共に、スタック領域として割当可能なメモリ領域が限られた中でも、実行可能な若しくは登録可能なタスク数の制限を緩和することができるという効果がある。

【0088】更に、それらタスクの実行開始から終了までを1単位として共有スタック領域に対するタスクの割り付けを排他的に制御することにより、複雑なメモリ制御が不要になり、しかも、プログラムサイズも小さくなり、処理の高速化も可能になる。

【0089】さらに、1チップ型のマイクロコンピュータのようにスタック領域として割当可能なメモリ領域が限られた中でも、管理すべきタスクを構成する各種ユーザープログラム若しくはアプリケーションプログラムに対してOSの柔軟な対応を可能にする。

【図面の簡単な説明】

【図1】図1は本発明に係るスタック制御方式をタスクの実行要求に際しての処理に着目した原理的な一例フローチャートである。

【図2】図2は本発明に係るスタック制御方式をタスクの実行終了に際しての処理に着目した原理的な一例フローチャートである。

【図3】図3は図1の処理によってタスクの実行開始が遅延された状態を示す一例説明図である。

【図4】図4はタスクによるスタック領域の共有関係を示す一例説明図である。

【図5】図5はマルチタスク制御OSとタスクとの一例関係説明図である。

【図6】図6はマルチタスク制御OSがタスクを管理するときの一例状態遷移図である。

【図7】図7は本発明に係る1チップマイクロコンピュータの実施例ブロック図である。

【図8】図8は図7に示されるマイクロコンピュータの一例アドレスマップである。

【図9】図9は図7に示されるマイクロコンピュータの内蔵ROMにおける一部のアドレスマップである。

【図10】図10は図7に示されるマイクロコンピュータの内蔵RAMのアドレスマップである。

【図11】図11はタスク管理テーブルの一例説明図である。

【図12】図12はタスク、スタック領域、及びスタック制御領域の一例関係説明図である。

【図13】図13はタスクの実行要求に際してのスタック領域に対するタスク割付制御の一例フローチャートである。

【図14】図14は実行開始が遅延されたタスクの取扱いとスタック領域の解放を考慮してタスクの実行終了に着目したタスク割付制御の一例フローチャートである。

【図15】図15はタスクの実行状態におけるスタック領域の一例説明図である。

【図16】図16はCPUのOS実行状態とタスク実行状態のタイムチャートを示す一例説明図である。

【符号の説明】

OS オペレーティングシステム

P1乃至P11 タスク

A乃至E スタック領域

10 マイクロコンピュータ

11 中央処理装置

12 ROM

13 RAM

14 I/Oインタフェース

15 タイマ

SPD1乃至SPD11 スタック指示データ

40 TCB1乃至TCB11 タスク管理テーブル

Ad乃至Ed スタック管理変数

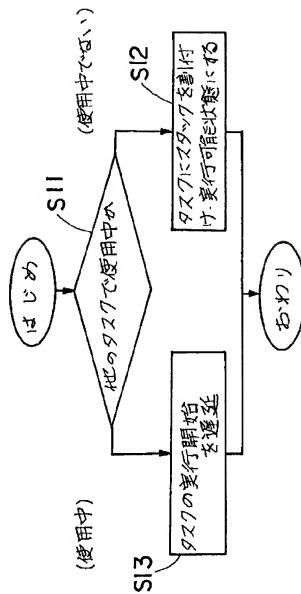
Ap乃至Ep スタック行列管理ポインタ

21 ポインタ群

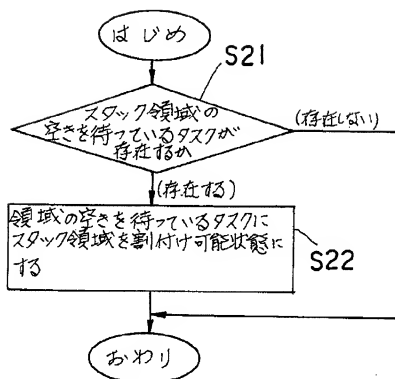
24 実行待ち行列管理ポインタ

【図1】

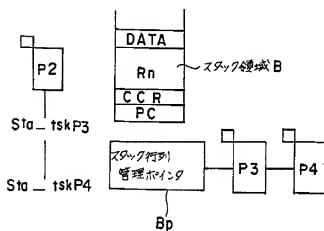
【 図 1 】



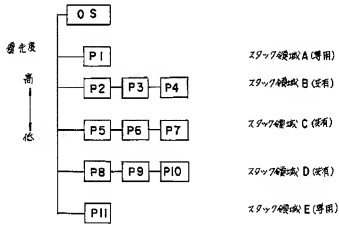
【図2】



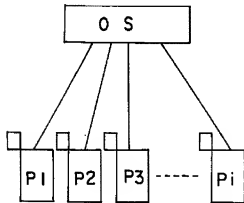
【図3】



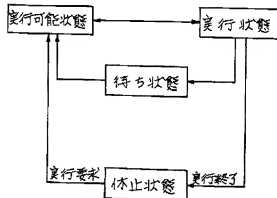
【図4】



【図5】



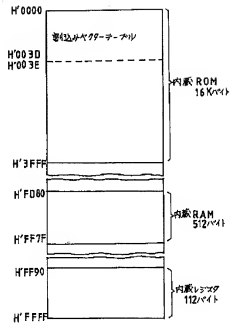
【図6】



【図7】

【図8】

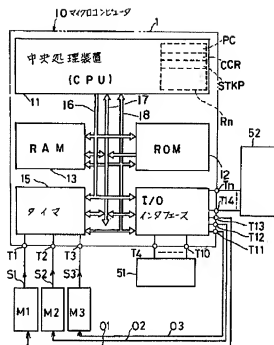
【図8】



【図9】

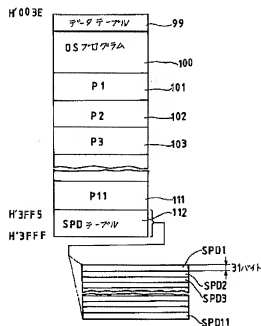
【図7】

【図7】



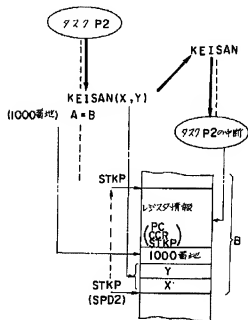
【図9】

【図9】



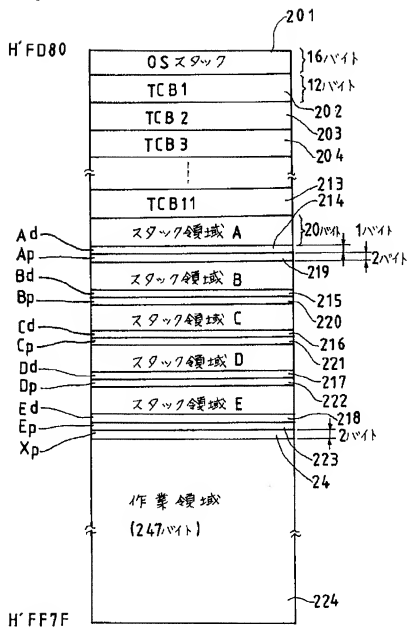
【図15】

【図15】



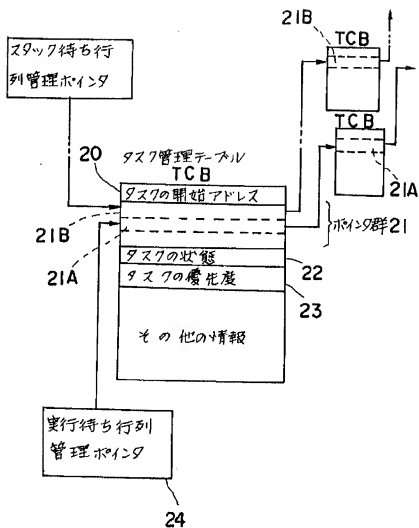
【図10】

【図10】



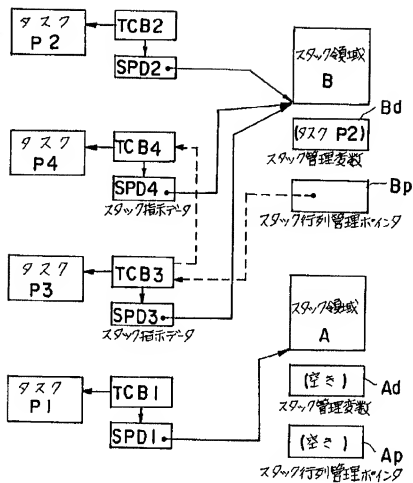
【図11】

【図11】



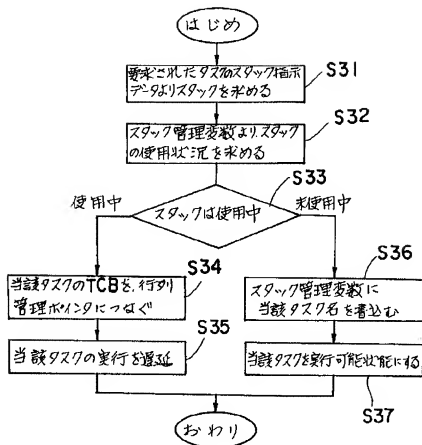
【図12】

【 図 1 2 】



【図13】

【 図 1 3 】



【図14】

【 図 1 4 】

